Application No.: 09/867,791

OIPE JUN 1 3 2005 PATENT & TRADEMARK OFFICE JC69

Docket No.: 13220/021001; P5834

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
David Boreham et al.

Conf. No.: 7719

Application No.: 09/867,791

Art Unit: 2167

Filed: May 29, 2001

Examiner: K. S. Lu

For: METHOD AND SYSTEM FOR SHARING ENTRY ATTRIBUTES IN A DIRECTORY SERVER USING CLASS OF SERVICE

DECLARATION OF KIERAN CROSS ROWLEY

1. I, Kieran Cross Rowley, was a technical writer at Netscape Communications Corp. (which was part of the iPlanet alliance between Sun Microsystems, Inc. and Netscape Communications Corp.) and wrote the Sun Microsystems, Inc. "iPlanet Directory Server Administrator's Guide, Version 5.0," published April 2001 (hereafter "iPlanet").

2. I, Kieran Cross Rowley, received all the technical content of iPlanet from David W. Boreham, Peter Rowley, and Mark C. Smith.

3. I, Kieran Cross Rowley, used the document written by David W. Boreham and Peter Rowley entitled "Netscape Directory Server 5.0 Class of Service Facility" to write iPlanet. The aforementioned document is attached as Appendix A.

4. I, Kieran Cross Rowley, had numerous meetings with David W. Boreham, Peter Rowley, and Mark C. Smith to discuss and review the technical content of the Class of Service Facility described in iPlanet.

I, Kieran Cross Rowley, hereby declare that all statements made herein of my own knowledge are true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Signed this day 10, of June 2005.

Kieran C Rowley
Kieran Cross Rowley

Appendix A
Netscape Directory Server 5.0 Class of Service Facility

Netscape Directory Server 5.0 Class of Service Facility

David Boreham, Pete Rowley, Netscape Directory Team
August 20, 1999

Introduction

Motivated by the desire to make management easier and to reduce storage requirements, many applications have some kind of "class of service" concept. The idea being that properties considered to belong to individual users' profiles are in fact determined as a group by the user's "class of service". Consider for example a voice mail system. Each user is subject to a number of resource limits including the number of messages permitted in their mailbox; the maximum length of message which may be recorded; whether or not they're allowed to make outgoing calls via the automated attendant. Rather than storing the individual values for each of these resource limits on each users' directory entry, several classes of service are defined, each with resource limits appropriate to a different type of user. The "superuser" class might have very high resource limits while the "ordinary user" class would have lower limits. Now each user's directory entry need only contain the service class to which they belong. This is both smaller than the complete set of resource limit attributes, and also much easier to manage because the number of service classes is much smaller than the number of users.

Typically the class of service feature is implemented by the application and not by the storage system which contains the user profile data (e.g. a database or directory server). This however is more by necessity than design, since these storage systems usually don't offer any class of service capability.

This document describes a class of service facility for Netscape Directory 4.x, to be implemented in a plugin.

How it works

Clients of the Directory Server such as a Messaging Server, read user's entries in the normal way. They see attributes on these entries as usual. Some attribute values may not have been stored with the entry however, they have been generated by class of service logic as the entry is sent to the client. The values returned for these attributes are determined by :

- The entry's DN. (different class of service regimes may be maintained for different portions of the DIT).
- A service class attribute value stored with the entry (for example: "nsmail-cos:ordinary"). The absence of the attribute altogether can also imply a specific default class of service.
- The attribute values stored in a class template entry. One such entry is selected to supply the attribute values for one particular service class.
- The objectclass of the entry. Class of service attribute values will only be generated where the entry has an objectclass which allows the attribute. If schema checking is turned off, this rule is not enforced.
- The attribute values stored in some paticular entry in the DIT, for example a container entry.
- The attribure values stored in an entry pointed to by a DN stored in an attribute of the entry.

Note that multiple class of service schemes may be defined within a server. It is illegal to define schemes such that they conflict with each other.

The class template entries for a given class of service scheme are all stored in the DIT, as children of a common parent entry. The relative DN's of the template entries are the respective values of the service class attribute, plus the special RDN "*cosSpecifier*-default", where *cosSpecifier* is the value of the cosSpecifier attribute in the cosDefinition, to be used when the service class attribute value is NULL or not present. For example, a cosDefinition with a cosSpecifier value of "test", would have a default template of "test-default".

Class of Service Definition

Classes of service definitions (that is, all the information, save the template entries, needed in order to generate attribute values defined by a class of service) are stored in LDAP Subentries, which can be located anywhere in the DIT. Class of Service definition entries have objectclass cosClassicDefinition, cosPointerDefinition or cosIndirectDefinition (cosDefinition is supported to provide compatibility with the DS4.1 COS plugin):

Classic COS, where template entries are selected based on the value of a specifier attribute in the entry:

```
dn: cn=messaging_server_cos, ou=people, o=acme
objectclass: top
objectclass: LDAPSubentry
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=LocalConfig, cn=MailServerCOS
cosSpecifier: mailServiceClass
cosAttribute: mailboxquota
cosAttribute: maysendexternalmail
```

In the example above we define a class of service specified by the attribute "mailServiceClass". This service class set defines the values of attributes "mailboxquota" and "maySendExternalMail". The values are to be taken from template entries stored in the DIT under " cn=LocalConfig, cn=MailServerCOS". The service classes only pertain to entries within the portion of the tree under "ou=People, o=Netscape.com".

Here's an example of two corresponding template entries:

```
dn: cn=user, cn=MailServerCOS, cn=LocalConfig, o=NetscapeRoot
objectclass: top
objectclass: mailserveruser
mailboxquota: 10000000
maysendexternalmail: TRUE

dn: cn=guest, cn=MailServerCOS, cn=LocalConfig, o=NetscapeRoot
objectclass: top
objectclass: mailserveruser
mailboxquota: 1000000
maysendexternalmail: FALSE
```

Pointer COS, where the definition points to a single template entry:

```
dn: cn=messaging_server_cos, ou=people, o=acme
objectclass: top
objectclass: LDAPSubentry
objectclass: cosSuperDefinition
```

```
objectclass: cosPointerDefinition
cosTemplateDn: ou=people, o=acme
cosAttribute: faxscimilieTelephoneNumber default
```

In the example above, every person's entry will inherit the organization's fax number, unless their entry has its own fax number attribute.

Indirect COS, where a DN-valued attribute in the entry points to the template entry:

```
dn: cn=messaging_server_cos, ou=people, o=acme
objectclass: top
objectclass: LDAPSubentry
objectclass: cosSuperDefinition
objectclass: cosIndirectDefinition
cosIndirectSpecifier: manager
cosAttribute: accountingCode
```

In the example above, each person's entry will inherit the "accounting code" from their manager's entry. Thus accounting codes automatically change to the correct new value when an employee moves departments.

Interaction with Stored Attribute Values

If the class of service logic detects that the attribute who's value it is generating is in fact already stored on the entry, the default action is to supply the stored value to the client. However, the server's behaviour can be controlled by means of the cosDefinition entry. The cosAttribute values allow an additional qualifier appended after the attribute type name. Valid qualifier values are "override" and "default". An absent qualifier means the same as "default". Override means that the server will always return the value generated by the class of service logic even when there is a value stored with the entry. Default means that the server will only return a generated value if there is no corresponding attribute value stored with the entry.

e.g.:

```
dn: cn=messaging_server_cos, cn=cosdefinitions, o=NetscapeRoot
objectclass: top
objectclass: cosSuperDefinition
objectclass: cosClassicDefinition
cosTemplateDn: cn=LocalConfig, cn=MailServerCOS
cosSpecifier: mailServiceClass
cosAttribute: mailboxquota override
cosAttribute: maysendexternalmail default
```

Configuration and Management

Because all the configuration information and template data is stored as entries in the directory, standard LDAP tools can be used for configuration and management. Specialized scripts, command-line tools and graphical UI could be developed. These would use the LDAP SDK to inspect and change the configuration. Further discussion of such tools is outside the scope of this document.

Access Control

The server controls access to attributes generated by a service class in exactly the same way as regular stored attributes.
Access control rules depending upon the value of attributes within an entry will behave as expected even when those attribute values are generated by COS.

Implementation Restrications

Tthere are some hopefully insignificant restrictions that should be noted:

- LDAP search requests containing a filter that references an attribute defined by class of service may not be serviced. The server will respond "unwilling to perform" when presented with such an unsupported request.